

# BEST AVAILABLE COPY PATENT ABSTRACTS OF JAPAN

(11)Publication number : 07-093269

(43)Date of publication of application : 07.04.1995

(51)Int.Cl.

G06F 15/16  
G06F 9/45

(21)Application number : 05-292919

(71)Applicant : FUJITSU LTD

(22)Date of filing : 24.11.1993

(72)Inventor : IWASHITA HIDETOSHI

(30)Priority

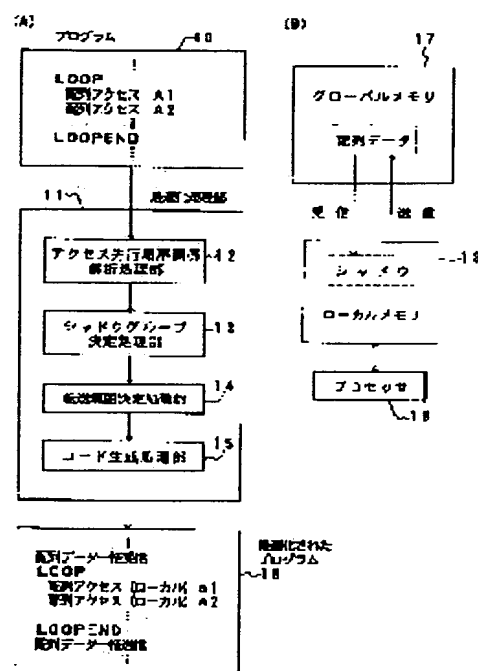
Priority number : 05185065 Priority date : 27.07.1993 Priority country : JP

## 54) BATCH PROCESSING METHOD FOR DATA TRANSFER

## 57)Abstract:

**PURPOSE:** To reduce overhead and to reduce a data transfer amount by classifying arrangement access inside a loop into groups based on preceding order relation and generating batch access corresponding to each group outside the loop.

**CONSTITUTION:** When there is access A1, A2... of arrangement data at a global memory 17 concerning a program 10 of an optimizing target, this access is classified into groups for making shadow common, and a transfer range and the shape of the shadow are decided. Then, an instruction for simultaneously receiving the required arrangement data from the global memory 17 all together for each of shadow is added before the loop, and an instruction for simultaneously transmitting the arrangement data as the processing result to the global memory 17 all together is added after the loop. Inside the loop, the shadow at a local memory 18 is accessed. Thus, it is not necessary for each loop to transfer the data of an element unit with the global memory 17, the number of times of data transfer is reduced due to batch transfer, and the transfer amount can be reduced.



## LEGAL STATUS

Date of request for examination] 20.12.1999

Date of sending the examiner's decision of rejection]

Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

Date of final disposal for application]

Patent number] 3261239

Date of registration] 14.12.2001

Number of appeal against examiner's decision of rejection]

Date of requesting appeal against examiner's decision of rejection]

Date of extinction of right]

Copyright (C); 1998,2003 Japan Patent Office

(19)日本国特許庁 (J P)

(12) 公 開 特 許 公 報 (A)

(11)特許出願公開番号

特開平7-93269

(43)公開日 平成7年(1995)4月7日

(51)Int.Cl.<sup>6</sup>

G 0 6 F 15/16  
9/45

識別記号

4 3 0 A 7429-5L

9292-5B

庁内整理番号

F I

G 0 6 F 9/ 44

3 2 2 G

技術表示箇所

審査請求 未請求 請求項の数 2 O L (全 13 頁)

(21)出願番号 特願平5-292919

(22)出願日 平成5年(1993)11月24日

(31)優先権主張番号 特願平5-185065

(32)優先日 平5(1993)7月27日

(33)優先権主張国 日本 (J P)

(71)出願人 000005223

富士通株式会社

神奈川県川崎市中原区上小田中1015番地

(72)発明者 岩下 英俊

神奈川県川崎市中原区上小田中1015番地

富士通株式会社内

(74)代理人 弁理士 小笠原 吉義 (外2名)

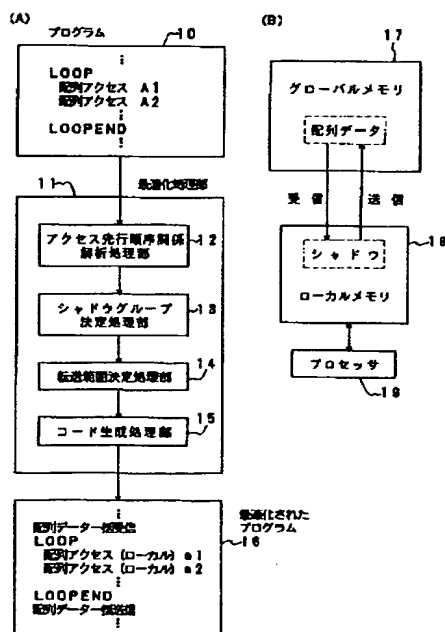
(54)【発明の名称】 データ転送の一括化処理方法

(57)【要約】

【目的】 メモリが遠近2階層にモデル化できる計算機システムにおけるデータ転送の一括化処理方法に関し、ループ内のデータ転送を一括化してループ外に出すことにより、転送回数および転送総量を削減することを目的とする。

【構成】 グローバルメモリ17にある配列データのアクセスを含むループ手続きを持つプログラム10について、そのアクセスの先行順序関係を解析し、シャドウを共通化するグループに分類する。そして、各グループに属するアクセスについて、それぞれアクセス要素の集合和を求め、データを転送する範囲を決定する。その範囲の配列データを一括アクセスする命令コードをループ外に生成し、最適化されたプログラム16を出力する。

本発明の原理説明図



## 【特許請求の範囲】

【請求項 1】 1 または複数のプロセッサ(19)と、そのプロセッサ(19)が高速にアクセスできる第 1 のメモリ(18)と、大容量または他のプロセッサと共用するデータを保持する第 2 のメモリ(17)とを備え、前記第 2 のメモリ(17)が保持するデータを前記第 1 のメモリ(18)に転送して処理する計算機システム上で動作するプログラム(10)を最適化する処理方法であって、前記第 2 のメモリ(17)にある配列データのアクセスを含むループ手続きが与えられたとき、そのアクセスの先行順序関係を解析する処理過程(12)と、アクセスの先行順序関係を解析した結果に基づいて、ループ内の配列アクセスを、前記第 1 のメモリ(18)におけるデータの一時的な格納場所の共通化が必要なグループに分類する処理過程(13)と、各グループに属するアクセスについて、それぞれアクセス要素の集合和を求め、データを転送する範囲を決定する処理過程(14)と、前記各グループに属する転送範囲の配列データを一括アクセスする命令コードをループ外に生成する処理過程(15)とを有することを特徴とするデータ転送の一括化処理方法。

【請求項 2】 1 または複数のプロセッサ(19)と、そのプロセッサ(19)が高速にアクセスできる第 1 のメモリ(18)と、大容量または他のプロセッサと共用するデータを保持する第 2 のメモリ(17)とを備え、前記第 2 のメモリ(17)が保持するデータを前記第 1 のメモリ(18)に転送して処理する計算機システム上で動作するプログラム(10)を最適化する処理方法であって、前記第 2 のメモリ(17)にある配列データのアクセスを含むループ手続きが与えられたとき、そのアクセスの先行順序関係を解析する処理過程(12)と、アクセスの先行順序関係を解析した結果に基づいて、ループ内の配列アクセスを、前記第 1 のメモリ(18)におけるデータの一時的な格納場所の共通化が必要なグループに分類する処理過程(13)と、各グループに属するアクセスについて、ループ内で参照するアクセス要素の集合うち、参照に先立って定義されるアクセス要素の集合を受信転送範囲から除外し、各グループごとに受信転送範囲および送信転送範囲についてのアクセス要素の集合和を求め、データを転送する範囲を決定する処理過程(14)と、前記受信転送範囲の配列データを一括アクセスする命令コードをループの前に生成し、前記送信転送範囲の配列データを一括アクセスする命令コードをループの後に生成する処理過程(15)とを有することを特徴とするデータ転送の一括化処理方法。

## 【発明の詳細な説明】

## 【0001】

【産業上の利用分野】 本発明は、階層型のメモリを持つ並列計算機等において、遠方の配列データのアクセスを含むループ手続きが与えられたとき、アクセスを一括化してループ外に出すことにより、プログラムの最適化を実現したデータ転送の一括化処理方法に関する。

【0002】 並列計算機、ネットワーク接続された分散計算機環境、大容量外部メモリ装置を使用する計算機等では、一般に遠近 2 階層にモデル化されるメモリを使用する。このような階層型のメモリシステムにおいて、オーバーヘッドの削減のためには、メモリ間のデータ転送回数および全体的なデータ転送量を減らすことが必要となる。

## 【0003】

【従来の技術】 図 13 は本発明に関係する 2 階層メモリのモデル説明図、図 14 は本発明の課題説明図である。

【0004】 並列計算機、ネットワーク接続された分散計算機環境、大容量外部メモリ装置を使用する計算機等では、図 13 に示すようにメモリが遠近 2 階層にモデル化できる。図中の 80 はプロセッサ、81 はプロセッサ 80 が高速に直接アクセスできるローカル(Local)メモリ空間、82 はローカルメモリ空間 81 を介してアクセスされるデータを保持するグローバル(Global)メモリ空間である。グローバルメモリ空間 82 は、仮想的なものであってもよく、また、例えば分散メモリ型並列計算機では、物理的には他のプロセッサのローカルメモリであってもよい。

【0005】 図 13 に示すような計算機構成では、グローバルメモリ空間 82 とローカルメモリ空間 81 との間のデータ転送は、1 要素ずつ行うよりも一括化して転送回数を減らすほうが、立ち上がりのオーバーヘッドが削減され効率がよい。また、可能な限り転送量を減らすほうがよい。

【0006】 しかし、従来、例えば図 14 (a) に示すようなプログラムのループにおいて、グローバルメモリ空間にある配列データ A(i) 等を参照または定義する命令があるような場合に、グローバルメモリ空間 82 とローカルメモリ空間 81 との間で必要最小限のデータだけを一括して転送することは考えられていなかった。そのため、データ転送回数が多くなる、または転送総量が多くなるという問題があった。なお、以下の説明において、グローバルメモリ空間 82 にある配列データは英大文字で始まる配列名〔例：A(i)〕で表し、ローカルメモリ空間 81 上の配列データは英小文字で始まる配列名〔例：c(i)〕で表す。

## 【0007】

【発明が解決しようとする課題】 ところで、ループにおけるデータをできるだけ大きな単位で扱う技術として、いわゆるベクトル計算機における最適化処理方法がある。ここで用いられるベクトル化技術では、アクセスするデータの先行順序関係を解析して、できるだけ大量なデータのベクトル計算を効率よく実行できるような命令コードの生成を行っている。

【0008】 前述のように、グローバルメモリ空間 82 とローカルメモリ空間 81 との間のデータ転送は、1 要素ずつ行うよりも一括化して転送回数を減らすほうが、

3

立ち上がりのオーバーヘッドが削減され効率がよい。そこで、ベクトル化技術を流用することを考える。ベクトル化技術を流用してデータの一括転送を図ることにより、2階層メモリにモデル化されるシステムにおいても、ある程度のデータ転送の一括化は可能であると考えられる。

【0009】例えば、図14(a)に示すようなプログラムをベクトル化すると、図14(b)に示すようになる。図14(b)において、c(1:100)は、ローカルメモリ空間81における配列cの1から100までの要素を表す。A(1:100)は、グローバルメモリ空間82における配列Aの1から100までの要素を表す。他の配列表記も同様である。

【0010】まず、図14(a)に示すプログラムを、図14(b)に示すように既存の方法でベクトル化し、これをもとに、図14(c)に示すようにデータの受信/送信命令を付加する。すなわち、グローバル配列Aの参照を、グローバルメモリ空間82からローカルメモリ空間81の一時域(tmp1)への転送(receive)と、一時域(tmp1)の参照に置き換え、また、グローバル配列Aの定義を、ローカルメモリ空間81における一時域(tmp2)の定義と、一時域(tmp2)からグローバルメモリ空間82への転送(send)に置き換える。

【0011】しかし、この方法では、データ依存関係によってベクトル化できない場合には、データ転送の一括化はできない。例えば図14(d)に示すようなプログラムの場合である。このプログラムでは、配列A(1)の参照と配列A(2)の定義が同じループ内で行われ、次のループで配列A(2)が参照される。したがって、配列A(2)の定義が、配列A(2)の参照より先に実行されるが、ベクトル化するとこの順序が逆転することになるので、ベクトル化はできない。すなわち、この例の場合、一時域の適当な共通化によって、転送の一括化は可能であると考えられるが、ベクトル化技術の延長では実現は難しい。

【0012】また、ベクトル化では、主に単一のループをその対象とするので、データ転送の効率化に十分な要素量の一括化ができない。一般にベクトル化では数十要素で十分であるが、データ転送では数千〜数万要素を一括化してはじめて十分な効率が得られることが多い。特に、ループが多重の場合、多重ループ全体について一括化する技術が必要とされる。

【0013】本発明は、一時変数をグループ分けし、グループごとに共通化することにより、ベクトル化できないループについても転送の一括化を可能にすることを目的とする。また、多重ループについてもデータ転送の一括化を図り、プログラムの高速実行を可能とすることを目的とする。さらにまた、転送データ量の削減を図ることを目的とする。

【0014】

4

【課題を解決するための手段】図1は本発明の原理説明図である。図1(A)に示すプログラム10は、図1(B)に示すような計算機システムで実行されるプログラムであって、グローバルメモリ17が保持する配列データのアクセスを含むループ(LOOP)手続きを持つプログラムである。プロセッサ19は、グローバルメモリ17の配列データを通常の演算命令でダイレクトに読み込んで処理したり、グローバルメモリ17へダイレクトに書き出したりすることはできず、一旦ローカルメモリ18に転送して、ローカルメモリ18内で処理し、書き出す際には、処理結果をローカルメモリ18からグローバルメモリ17へ転送する。ローカルメモリ18における、グローバルメモリ17の写しをシャドウという。

【0015】最適化処理部11は、プログラム10を最適化し、実行の高速化を図る手段である。本発明では、各処理部により以下の処理を行う。アクセス先行順序関係解析処理部12は、グローバルメモリ17にある配列データのアクセスを含むループ手続きを持つプログラム10が与えられたとき、そのアクセスの先行順序関係を解析する。

【0016】シャドウグループ決定処理部13は、アクセスの先行順序関係を解析した結果に基づいて、ループ内の配列アクセスを、シャドウを共通化するグループ(これを、シャドウグループという)に分類する。すなわち、先行順序が特定の関係にある複数の配列アクセスは必ず同一のグループに属するように分類する。

【0017】転送範囲決定処理部14は、各シャドウグループに属するアクセスについて、それぞれアクセス要素の集合和を求め、データを転送する範囲を決定する。特に、請求項2記載の発明では、転送範囲決定処理部14は、ループ内で参照するアクセス要素の集合のうち、参照に先立って定義されるアクセス要素の集合を受信転送範囲から除外して、受信転送範囲および送信転送範囲についてのアクセス要素の集合和を求め、データを転送する範囲を決定することにより、さらに受信転送データ量の削減を図る。

【0018】コード生成処理部15は、各シャドウグループに属する転送範囲の配列データを一括アクセスする命令コードをループ外に生成し、ループ内にあるグローバルメモリ17への配列アクセスは、シャドウのアクセスに変形する。これより、最適化されたプログラム16が生成され、出力される。

【0019】

【作用】本発明は、図1(B)のようなシステムにおいて、グローバルメモリ17に対するアクセスの一括化によるオーバーヘッドの削減と、データ転送量の削減を図ったものであり、そのために、ループ内の配列アクセスを先行順序関係を基にグループ分けし、各グループに対応する一括アクセスを、ループ外に生成するようにしたものである。

【0020】例えば、最適化対象のプログラム10において、ループ内にグローバルメモリ17における配列データのアクセスA1, A2, ...があるとき、これらをシャドウを共通化するグループに分類し、転送範囲とシャドウの形状を決定する。そして、シャドウごとにグローバルメモリ17から必要な配列データの一括受信を行う命令をループの前に付加し、また処理結果である配列データをグローバルメモリ17へ一括送信する命令をループの後に付加する。ループ内ではローカルメモリ18におけるシャドウをアクセスする。

【0021】これにより、各ループ実行ごとにグローバルメモリ17との間の要素単位のデータ転送が不要になり、一括転送によるデータ転送回数の削減および転送量の削減により、プログラムの高速実行が可能になる。ベクトル化できないようなケースでも、高速化を実現することができる。

【0022】さらに、請求項2記載の発明では、データ依存解析をもとに受信が不要である配列要素を算出し、それを受信対象から外すので、一層の転送データ量の削減が可能になる。

【0023】

【実施例】図2は、本発明の適用例説明図である。ソースファイル21は、最適化対象となるソースプログラムが格納されたファイルである。構文解析部22は、ソースファイル21からソースプログラムを読み出し、所定の構文規則に従って構文を解析し、解析結果を中間コード23として出力する。中間コード23は、本計算機システムによって定められた内部形式データであり、実質的内容はソースプログラムと同等のものである。中間コード23は、実際にファイルとして保持してもよく、またメモリ内に保持するようにしてもよい。

【0024】ここで、本発明に係る最適化処理部11による最適化を行う。前後に、他の最適化部28a, 28bによる最適化を行ってもよい。ファイル出力部24は、最適化処理後の中間コード23をもとに、最適化されたソースファイル25を出力する。これをコンパイラ26の入力ファイルとしてコンパイルすることにより、データの一括転送を行うオブジェクトモジュール27が生成される。

【0025】図2に示す方式は、いわゆるプリコンパイラ方式による最適化の構成例であるが、プリコンパイラ方式によらずに、コンパイラ26内部で最適化処理を実行するようにしてもよい。

【0026】〔請求項1の発明の実施例〕図3は、請求項1記載の発明の実施例処理フローチャートである。すべてのグローバル変数 $G_i$  ( $i=1, \dots, m$ ) について、図3に示す(a)~(c)を実行し、最後に(d)を実行する。ここでグローバル変数とは、図1(B)に示すグローバルメモリ17内に設けられる変数である。以下、この処理の流れに従って、本発明の実施例を説明する。

【0027】(a) 先行順序関係の解析

同一の対象配列をアクセスする組( $f, g$ )のうち、 $f, g$ ともに参照の場合を除くすべてについて、その先行順序関係を求める。先行順序関係は、次の4通りについて識別する。この先行順序関係の解析処理では、ベクトル化技術として知られている既存の技術を用いることができる。

【0028】

- ①  $f \rightarrow g$     アクセス $f$ がアクセス $g$ に常に先行する。
- ②  $g \rightarrow f$     アクセス $g$ がアクセス $f$ に常に先行する。
- ③  $f \leftrightarrow g$     アクセス $f$ がアクセス $g$ に先行するときに、逆のときがある。

【0029】または、順序関係が不明である。

- ④  $\phi$           アクセスされる配列要素に重なりがない。

(b) シャドウグループ $S_{ij}$  ( $j=1, \dots, n_i$ ) の決定  
対象配列のアクセスを、シャドウを共通化するグループ(シャドウグループ)に分類する。図4は、そのシャドウグループの決定論理説明図である。先行順序が図4に示す○印の関係にある2つのアクセスは、必ず同一のグループに属するように分類する。例えば、アクセス $f$ がアクセス $g$ に常に先行し、 $f$ が定義、 $g$ が参照のアクセスである場合、この2つのアクセスは同一のグループとする。

【0030】(c) シャドウ形状と転送範囲の決定

上記(b)で決定したすべてのシャドウグループ $S_{ij}$ について、以下の処理を行う。

【0031】① シャドウグループに属する参照および定義について、それぞれアクセス要素の集合和を求める。それがシャドウの受信および送信データの範囲となる。この受信データの範囲を $\Lambda$ 、送信データの範囲を $\Sigma$ で表す。

【0032】②  $\Lambda$ と $\Sigma$ の集合和を求める。これをシャドウ空間と呼ぶ。要素とシャドウ空間上の位置とのマッピングを示す関数かテーブルを求めておく。以下の説明では、このマッピングを $M$ で表す。

【0033】シャドウ空間が重なり合っていて、共通化するとメモリ領域が節約できると判断できた場合、複数のシャドウを共通化して一つのシャドウとしてもよい。そのとき、 $\Lambda$ および $\Sigma$ は、それぞれもとのシャドウの $\Lambda$ および $\Sigma$ の集合和とする。

【0034】(d) 出力コード生成

シャドウごとに、シャドウ空間と $\Lambda, \Sigma$ に応じたコード生成を行う。

① シャドウ空間を動的または静的に確保するコードを生成する。必要なシャドウ空間を包含する大きめの領域を確保してもよい。

【0035】② グローバルメモリからシャドウへ $\Lambda$ を受信するコードを、ループの直前に生成する。同様に、 $\Sigma$ をシャドウからグローバルメモリへ送信するコードを、ループの直後に生成する。

7

【0036】③ ループ内でのグローバルメモリへのアクセス部分は、マッピングMに応じて、シャドウのアクセスに変形する。次に、実際の最適化処理の例を説明する。図5は本発明の具体例説明図、図6は出力コードの例を示す図である。

【0037】例えば図5(A)に示すようなプログラムを最適化するものとする。このプログラムは、ループ変数J、Iについての多重ループとなっている。配列Aは、グローバル変数である。

#### 【0038】(a) 先行順序の解析

図5(A)に示す配列アクセスu1、u4をfとし配列アクセスu1~u5をgとして、前述した4通りのそれぞれの先行順序関係を調べると、図5(B)に示す先行順序関係の解析結果が得られる。例えば、アクセスu1とアクセスu2とは、それぞれA(I, J)の定義と、A(J, I)の参照であり、IとJの位置が逆転しているため、u1が先行するときとu2が先行するときがある。また、アクセスu1とアクセスu3とは、u3がA(I, J+30)であるため、u1のA(I, J)より常に先行する。

#### 【0039】(b) シャドウグループの決定

図5(B)に示す先行順序関係の解析結果を、図4に示す共通化の決定論理にあてはめると、共通化が必須のアクセスのペアは、次の組となる。

#### 【0040】① u1とu2

② u1とu5

③ u3とu4

よって、シャドウグループは、以下の2つとなる。

#### 【0041】① S1={u1, u2, u5}

② S2={u3, u4}

#### (c) シャドウ形状と転送範囲の決定

シャドウグループS1について、次のようにシャドウ空間を決定する。

#### 【0042】

$\Lambda = (u2 \text{ のアクセス範囲}) \cup (u5 \text{ のアクセス範囲})$

$= \{(1, 1), (2, 1), \dots, (100, 1), \dots, (100, 100)\}$

$\Sigma = (u1 \text{ のアクセス範囲})$

$= \{(1, 1), (2, 1), \dots, (100, 1), \dots, (100, 100)\}$

$\Lambda \cup \Sigma = \{(1, 1), (2, 1), \dots, (100, 1), \dots, (100, 100)\}$

この結果から、シャドウ領域として、ローカルメモリ上に二次元配列a1(100, 100)を確保する。グローバルとシャドウは、同じ添字を持つ要素同士を対応づける。すなわち、A(i, j)とa1(i, j)とを対応づける。

【0043】またシャドウグループS2について、次のようにシャドウ空間を決定する。

$\Lambda = (u3 \text{ のアクセス範囲})$

$= \{(1, 31), (2, 31), \dots, (100, 31), \dots, (100, 130)\}$

$\Sigma = (u4 \text{ のアクセス範囲})$

$= \{(1, 101), (2, 101), \dots, (100, 101), \dots, (100, 200)\}$

8

$\Lambda \cup \Sigma = \{(1, 31), (2, 31), \dots, (100, 31), \dots, (100, 200)\}$

この結果から、シャドウ領域として、二次元配列a2(100, 170)を確保する。グローバルとシャドウは、二次元目を30ずらして対応づける。すなわち、A(i, j)とa2(i, j-30)とを対応づける。

#### 【0044】(d) 出力コード生成

シャドウグループS1、S2について、シャドウ空間を確保するコードを生成し、グローバルメモリからシャドウへ $\Lambda$ を受信するコードと、シャドウからグローバルメモリへ $\Sigma$ を送信するコードを、それぞれループの前後に付加する。また、ループ内のアクセスを配列a1、a2へのアクセスに変換する。

【0045】この結果、図6に示すような内容のコードが生成されることになる。すなわち、図5(A)に示すプログラムが、図6に示すような最適化されたプログラムに変換される。

【0046】図6において、allocはローカルメモリの領域割り当て命令であり、「alloc a1(100, 100)」は、100×100要素の二次元配列a1をローカルメモリに確保することを指示している。同様に、「alloc a2(100, 170)」は、100×170要素の二次元配列a2をローカルメモリに確保することを指示している。

【0047】receiveは、グローバルメモリからローカルメモリへのデータ転送(受信)命令であり、「receive a1(1:100, 1:100)=A(1:100, 1:100)」は、二次元配列のA(1, 1)からA(100, 100)までの全データを、a1(1, 1)からa1(100, 100)までの領域に転送することを指示している。次の「receive a2(1:100, 1:100)=A(1:100, 31:130)」は、二次元配列のA(1, 31)からA(100, 130)までのデータを、a2(1, 1)からa2(100, 100)までの領域に転送することを指示している。

【0048】アクセスu1~u5の命令は、図6に示すように配列a1、a2の対応する領域に対するアクセスに変更される。sendは、ローカルメモリからグローバルメモリへのデータ転送(送信)命令であり、「send A(1:100, 1:100)=a1(1:100, 1:100)」は、二次元配列のa1(1, 1)からa1(100, 100)までのデータを、A(1, 1)からA(100, 100)までの領域に転送することを指示している。次の「send A(1:100, 101:200)=a2(1:100, 71:170)」は、二次元配列のa2(1, 71)からa2(100, 170)までのデータを、A(1, 101)からA(100, 200)までの領域に転送することを指示している。

【0049】「free a1」、「free a2」

は、それぞれローカルメモリ上の配列  $a_1$ ,  $a_2$  の領域を解放する命令である。

〔請求項2の発明の実施例〕図7は、請求項2記載の本発明の実施例処理フローチャートである。

【0050】すべてのグローバル変数  $G_i$  ( $i = 1, \dots, m$ ) について、図7に示す(a), (b), (e), (f) を実行し、最後に(d) を実行する。ここでグローバル変数とは、図1(B)に示すグローバルメモリ17内に設けられる変数である。以下、この処理の流れに従って、本発明の実施例を説明するが、(a), (b), (d) の処理については請求項1記載の実施例と同様であるので、説明を省略する。

#### 【0051】(e) 受信転送範囲の縮小

処理(b) で決定したシャドウグループ  $S_{ij}$  ( $j = 1, \dots, n_i$ ) の配列要素の集合のうち、受信転送範囲から除外することができるアクセス要素範囲を検出する。図8は、受信転送範囲の縮小の実施例の処理フローチャートである。

【0052】図8において、シャドウグループ  $S_{ij}$  に属する参照を  $r_k$  ( $k = 1, \dots, K$ )、シャドウグループ  $S_{ij}$  に属する定義を  $g_l$  ( $l = 1, \dots, L$ )、参照または定義  $f$  でアクセスされる配列要素の集合を  $\alpha(f)$  とし、すべての参照  $r$  について、以下の処理を行う。

【0053】① すべての参照  $r_k$  について、参照  $r_k$  の受信転送範囲  $\lambda(r_k)$  の初期値を  $\alpha(r_k)$  とする。

② 参照  $r_k$  と同一のシャドウグループに属し、先行順序が  $g_l \rightarrow r_k$  の関係にあるかどうかを検出する。

【0054】③  $\lambda(r_k)$  と  $\alpha(g_l)$  の集合差を改めて  $\lambda(r_k)$  とする。すべての定義  $g_l$  についてこれを繰り返す。すなわち、 $k = 1$  と初期化し(処理100)、処理108によって  $k$  を1ずつ増加させて、 $k$  が参照の個数  $K$  を超える(処理101)まで、以下の処理を行う。処理102により、 $k$  番目の参照  $r_k$  の受信転送範囲  $\lambda(r_k)$  の初期値を  $\alpha(r_k)$  とする。次に、同一のシャドウグループに属する  $l = 1$  から  $L$  までの定義  $g_l$  について、先行順序が先行順序が  $g_l \rightarrow r_k$  の関係にあるものを検出し、検出できたならば、 $\lambda(r_k)$  と  $\alpha(g_l)$  の集合差を求めて、 $\lambda(r_k)$  と置き換える(処理103~107)。

【0055】なお、他の実施例として、参照  $r$  と同一のシャドウグループに属し、 $g \leftrightarrow r$  の関係にある定義  $g$  を検出して、 $\lambda(r)$  と  $\alpha(g)$  の共通要素のうち、最初のアクセスが定義であるものをさらに  $\lambda(r)$  から削除するようにしてもよい。

#### 【0056】(f) シャドウ形状と転送範囲の決定

上記(e) の処理の後、すべてのシャドウグループ  $S_{ij}$  について、以下の処理を行う。

【0057】① シャドウグループに属するすべての参照  $r$  の受信転送範囲  $\lambda(r)$  の集合和をシャドウの受信

転送データの範囲とする。これを  $\Lambda$  と表すと、シャドウグループ  $S_{ij}$  の受信転送範囲  $\Lambda$  は、 $\Lambda = \lambda(r_1) \cup \lambda(r_2) \cup \dots \cup \lambda(r_K)$  となる。また、シャドウグループに属するすべての定義  $g$  のアクセス範囲  $\alpha(g)$  の集合和をシャドウの送信転送範囲とする。以下、これを  $\Sigma$  と表す。

【0058】②  $\Lambda$  と  $\Sigma$  の集合和を求める。これをシャドウ空間と呼ぶ。要素とシャドウ空間上の位置とのマッピングを示す関数  $\phi$  を求めておく。シャドウ空間が重なり合っていて、共通化するとメモリ領域が節約できると判断できた場合、複数のシャドウを共通化して一つのシャドウとしてもよい。そのとき、 $\Lambda$  および  $\Sigma$  は、それぞれもとのシャドウの  $\Lambda$  および  $\Sigma$  の集合和とする。

【0059】次に、実際の最適化処理の例を前述の図5を用いて説明する。例えば図5(A)に示すようなプログラムを最適化するものとする。このプログラムは、ループ変数  $J$ ,  $I$  についての多重ループとなっている。配列  $A$  はグローバル変数である。

#### 【0060】(a) 先行順序の解析

図5(A)に示す配列アクセス  $u_1$ ,  $u_4$  を  $f$  とし配列アクセス  $u_1 \sim u_5$  を  $g$  とし、前述した4通りのそれぞれの先行順序関係を調べると、図5(B)に示す先行順序関係の解析結果が得られる。例えば、アクセス  $u_1$  とアクセス  $u_2$  とは、それぞれ  $A(I, J)$  の定義と、 $A(J, I)$  の参照であり、 $I$  と  $J$  の位置が逆転しているため、 $u_1$  が先行するときと  $u_2$  が先行するときがある。また、アクセス  $u_1$  とアクセス  $u_3$  とは、 $u_3$  が  $A(I, J+30)$  であるので、 $u_1$  の  $A(I, J)$  より常に先行する。

#### 【0061】(b) シャドウグループの決定

図5(B)に示す先行順序関係の解析結果を、前述の図4に示す共通化の決定論理にあてはめると、共通化が必須のアクセスのペアは、次の組となる。

【0062】①  $u_1$  と  $u_2$

②  $u_1$  と  $u_5$

③  $u_3$  と  $u_4$

よって、シャドウグループは、以下の2つとなる。

【0063】①  $S_1 = \{u_1, u_2, u_5\}$

②  $S_2 = \{u_3, u_4\}$

#### (e) 受信転送範囲の縮小

シャドウグループ  $S_1$  についての受信転送範囲は、次のようになる。

【0064】参照  $u_2$  については、定義  $u_1$  との先行順序関係が  $u_1 \leftrightarrow u_2$  である。参照  $u_2$  に対し、 $u_1$  の要素の一部が先行するが、パターンが複雑と判断し無視する。受信範囲の縮小化はここでは行わない。

【0065】 $\lambda(u_2) = \alpha(u_2)$

$= \{(1, 1), (2, 1), \dots, (100, 1), \dots, (100, 100)\}$

参照  $u_5$  に対して、定義  $u_1$  が常に先行する。

11

【0066】 $\lambda(u5) = \alpha(u5) - \alpha(u1) = \phi$   
一方、シャドウグループS2については、次のようになる。参照u3に対して、定義u4が常に先行する。

【0067】 $\lambda(u3) = \alpha(u3) - \alpha(u4)$   
 $= \{(1, 31), (2, 31), \dots, (100, 31), \dots, (100, 100)\}$

(f) シャドウ形状と転送範囲の決定

シャドウグループS1について、次のようにシャドウ空間を決定する。

【0068】 $\Lambda = \lambda(u2) \cup \lambda(u5)$   
 $= \{(1, 1), (2, 1), \dots, (100, 1), \dots, (100, 100)\}$   
 $\Sigma = \alpha(u1)$

$= \{(1, 1), (2, 1), \dots, (100, 1), \dots, (100, 100)\}$   
 $\Lambda \cup \Sigma = \{(1, 1), (2, 1), \dots, (100, 1), \dots, (100, 100)\}$   
 この結果から、シャドウ領域として、ローカルメモリ上に二次元配列a1(100, 100)を確保する。グローバルとシャドウは、同じ添字を持つ要素同士を対応づける。すなわち、A(i, j)とa1(i, j)とを対応づける。

【0069】またシャドウグループS2について、次のようにシャドウ空間を決定する。

$\Lambda = \lambda(u3)$   
 $= \{(1, 31), (2, 31), \dots, (100, 31), \dots, (100, 100)\}$   
 $\Sigma = \alpha(u4)$   
 $= \{(1, 101), (2, 101), \dots, (100, 101), \dots, (100, 200)\}$   
 $\Lambda \cup \Sigma = \{(1, 31), (2, 31), \dots, (100, 31), \dots, (100, 200)\}$

この結果から、シャドウ領域として、二次元配列a2(100, 170)を確保する。グローバルとシャドウは、二次元目を30ずらして対応づける。すなわち、A(i, j)とa2(i, j-30)とを対応づける。

【0070】(d) 出力コード生成

シャドウグループS1, S2について、シャドウ空間を確保するコードを生成し、グローバルメモリからシャドウへ $\Lambda$ を受信するコードと、シャドウからグローバルメモリへ $\Sigma$ を送信するコードを、それぞれループの前後に付加する。また、ループ内のアクセスを配列a1, a2へのアクセスに変換する。

【0071】この結果、図9に示すような内容のコードが生成されることになる。すなわち、図5(A)に示すプログラムから、図9に示すような最適化されたプログラムが得られる。

【0072】前述の実施例における図6の出力コードとの違いは、シャドウグループS2についての受信転送範囲 $\Lambda$ が、図6では二次元配列のA(1, 31)からA(100, 130)までの領域であるのに対し、本実施例では、A(1, 31)からA(100, 100)までの領域であり、受信転送範囲が縮小されていることである。

【0073】なお、上記(e)の受信転送範囲の縮小処理において、シャドウグループS1とS2を共通化する方法がよいと判断できた場合には、シャドウ空間を次のよう

12

にしもよい。

【0074】 $\Lambda = \Lambda(S1) \cup \Lambda(S2)$   
 $= \{(1, 1), (2, 1), \dots, (100, 1), \dots, (100, 100)\}$   
 $\Sigma = \Sigma(S1) \cup \Sigma(S2)$   
 $= \{(1, 1), (2, 1), \dots, (100, 1), \dots, (100, 200)\}$   
 $\Sigma \cup \Lambda = \{(1, 1), (2, 1), \dots, (100, 1), \dots, (100, 200)\}$

この結果から、シャドウ領域として、ローカルメモリ上に二次元配列a(100, 200)を確保する。グローバルとシャドウは、同じ添字を持つ要素同士を対応づける。すなわち、A(i, j)とa(i, j)とを対応づける。

【0075】この結果、図10に示すような内容のコードが生成されることになる。すなわち、受信転送範囲 $\Lambda$ は、 $\Lambda(S1) \cup \Lambda(S2)$ により、二次元配列のA(1, 1)からA(100, 100)までの領域が対象となる。また、送信転送範囲 $\Sigma$ は、 $\Sigma(S1) \cup \Sigma(S2)$ により、A(1, 1)からA(100, 200)までの領域が対象となる。

【0076】図11は、本発明の適用結果の例を示す図である。例えば図11(A)に示すようなプログラムについて、請求項1記載の発明によってデータ転送の一括化を図った場合、出力コードは図11(B)に示すようになる。また、請求項2記載の発明によって、さらに受信転送範囲の縮小化を行った場合、その結果は図11(C)に示すようになる。

【0077】図11(A)に示す元のプログラムから明らかのように、A(k) (ただし、k=2, 3, 4, ..., )の定義は、A(k)の参照より先行する。したがって、A(2), A(3), ...の受信は不要であり、図11(C)の例では、グローバルメモリから受信するデータがA(1)だけになっている。

【0078】図12は、本発明が適用可能なシステム構成の例を示す図である。本発明によるデータ転送の一括化処理方法は、例えば図12(A)~(C)に示すような2階層のメモリ構造を持つシステムで動作するプログラムを対象として適用することができる。

【0079】図12(A)のシステムは、共有メモリ4を各プロセッサ・エレメント1が共有する並列計算機システムである。各プロセッサ・エレメント1は、CPU2とローカルメモリ3からなる。CPU2が実行するプログラムのループ中に、共有メモリ4上の配列に対するアクセスがあるとき、本発明の適用により、その配列データをループの直前で共有メモリ4からローカルメモリ3へ受信するようにし、ループの直後でローカルメモリ3から共有メモリ4へ送信するようにすることにより、プログラムの高速化が可能になる。

【0080】図12(B)のシステムは、仮想的共有メモリを持つ並列計算機または分散計算機環境のシステムであって、各プロセッサ・エレメント1は、ネットワーク6で接続されている。各プロセッサ・エレメント1



は、CPU2とローカルメモリ3と他のプロセッサがアクセスできるメモリ5を持つ。各プロセッサ・エレメント1に分散して配置されたメモリ5は、例えば全体で一つのメモリ空間を構成するように扱われる。CPU2は、メモリ5のデータにアクセスする必要があるとき、メモリ5のデータをローカルメモリ3に複写して、ローカルメモリ3上でアクセスする。このような他のプロセッサがアクセスできるメモリ5を、図1に示すグローバルメモリ17として、本発明を適用することができる。

【0081】図12(C)のシステムは、大容量外部メモリ装置7を持つ計算機システムである。プロセッサ・エレメント1は、CPU2とローカルメモリ3とを持ち、大容量外部メモリ装置7上のデータを参照・更新する場合には、ローカルメモリ3にデータを複写してアクセスする。このようなローカルメモリ3と大容量外部メモリ装置7からなる2階層メモリのシステムにも、同様に本発明を適用し、プログラム実行の高速化を図ることができる。

【0082】

【発明の効果】以上説明したように、本発明によれば、アクセスの一括化により転送回数および転送総量の削減が可能になり、データ転送に関するオーバーヘッドが削減される。また、対象変数の先行順序関係のみで転送対象を共通化して抽出することができるので、従来のベクトル化技術におけるベクトル可否等の判断よりも容易であり、効果も大きい。また、転送を多重ループの外に出すことができるため、さらに転送のスケジューリングなどの最適化が可能になる。例えば、非同期転送が可能なハードウェアであれば、受信転送をなるべく早く開始し、送信転送の終了待ちをなるべく遅らせることにより、計算と転送のオーバーラップによる処理の高速化を図ることができるようになる。

【図4】

シャドウグループの決定論理説明図

	f:参照 g:定義	f:定義 g:参照	f:定義 g:定義
f→g	—	○	○
f←→g	○	○	○
g→f	○	—	○
φ	—	—	—

【図面の簡単な説明】

【図1】本発明の原理説明図である。

【図2】本発明の適用例説明図である。

【図3】本発明の実施例処理フローチャートである。

【図4】シャドウグループの決定論理説明図である。

【図5】本発明の具体例説明図である。

【図6】本発明の実施例による出力コードの例を示す図である。

【図7】本発明の実施例処理フローチャートである。

【図8】受信転送範囲縮小の実施例処理フローチャートである。

【図9】本発明の実施例による出力コードの例を示す図である。

【図10】本発明の実施例による出力コードの例を示す図である。

【図11】本発明の適用結果の例を示す図である。

【図12】本発明が適用可能なシステム構成の例を示す図である。

【図13】2階層メモリのモデル説明図である。

【図14】本発明の課題説明図である。

【符号の説明】

- 10 プログラム
- 11 最適化処理部
- 12 アクセス先行順序関係解析処理部
- 13 シャドウグループ決定処理部
- 14 転送範囲決定処理部
- 15 コード生成処理部
- 16 最適化されたプログラム
- 17 グローバルメモリ
- 18 ローカルメモリ
- 19 プロセッサ

【図5】

具体例説明図

(A)

```

DO J=1,100
DO I=1,100
  A(I,J)= ...
  ... =A(J,I)
  ... =A(I,J+30)
  A(I,J+100)= ...
  ... =A(I,1)
END DO
END DO

```

u1  
u2  
u3  
u4  
u5

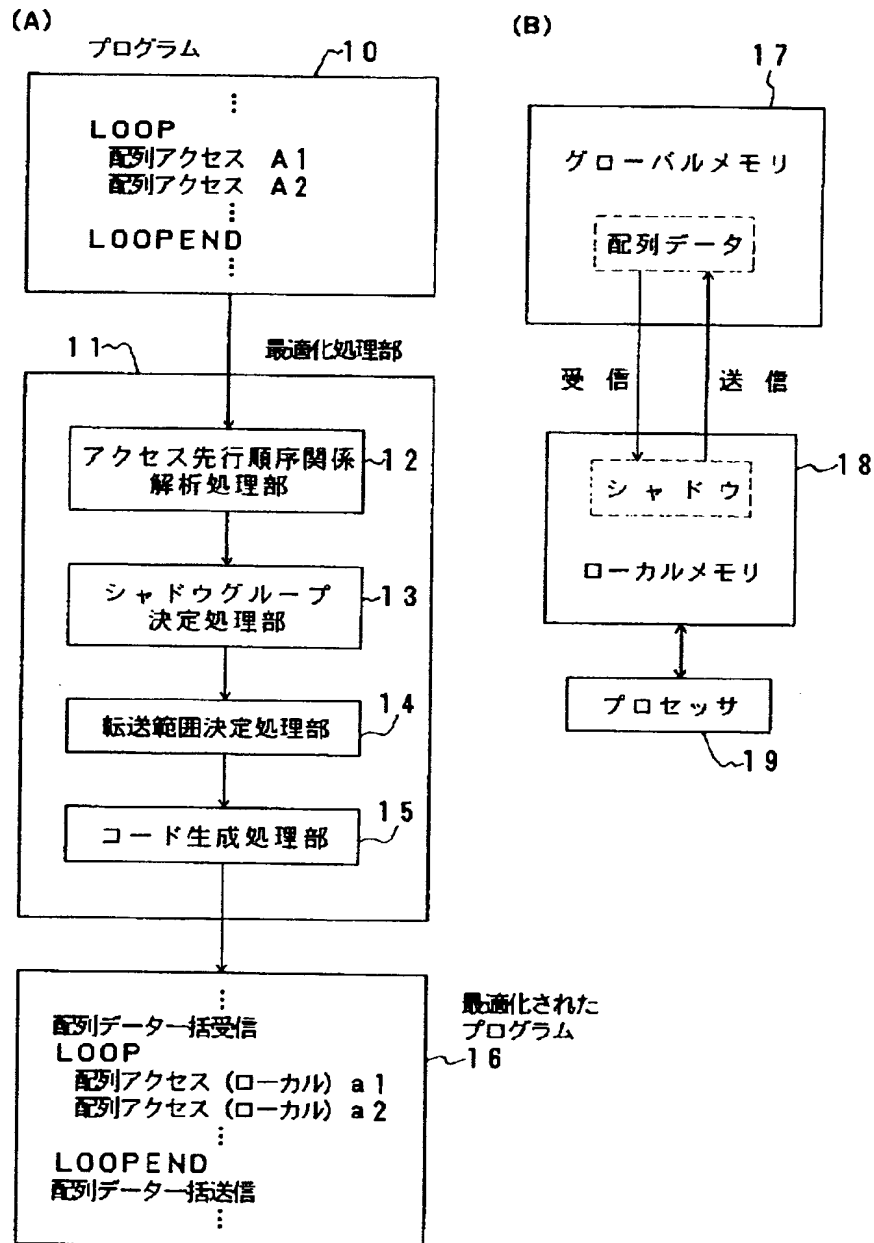
(B)

先行順序関係の解析

f \ g	u1	u2	u3	u4	u5
u1	φ	f←→g	g→f	φ	f→g
u4	φ	φ	f→g	φ	φ

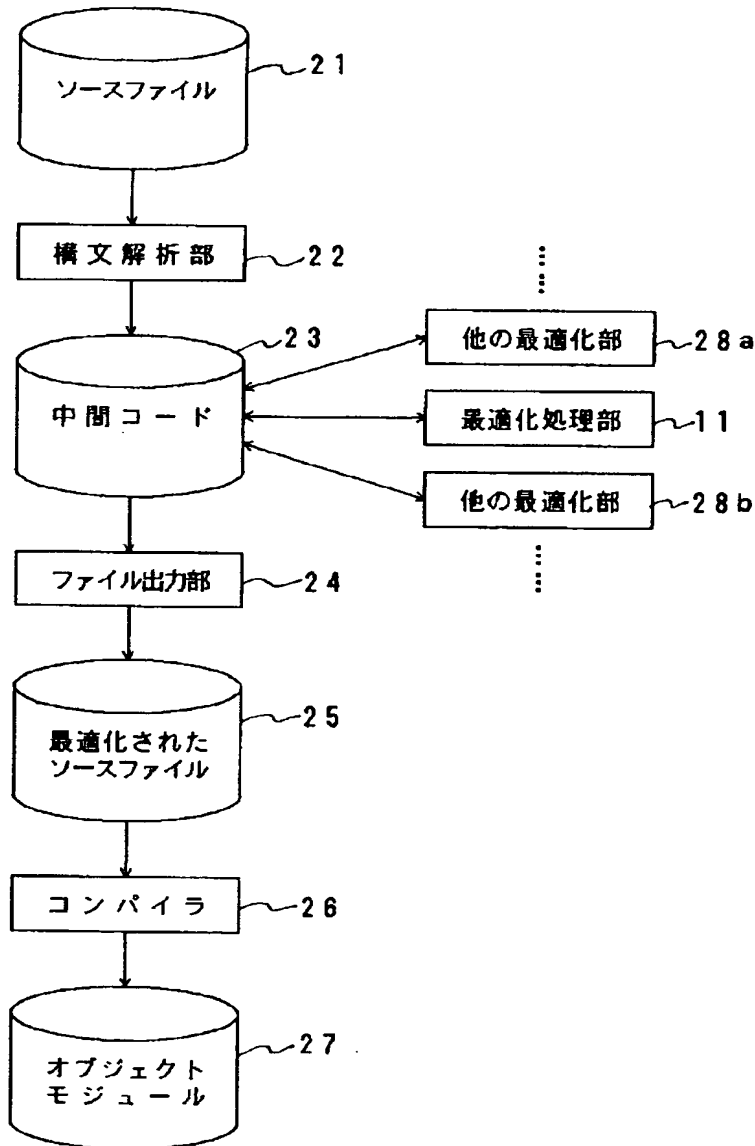
【図1】

## 本発明の原理説明図



【図 2】

## 本発明の適用例説明図



【図 1 1】

## 本発明の適用結果の例

- (A)
- ```

DO i=1,100
  c(i)=A(i)+b(i)
  A(i+1)=c(i)
END DO

```
- (B)
- ```

receive tmp(1:100)=A(1:100)
DO i=1,100
  c(i)=tmp(i)+b(i)
  tmp(i+1)=c(i)
END DO
send A(2:101)=tmp(2:101)

```
- (C)
- ```

receive tmp(1:1)=A(1:1)
DO i=1,100
  c(i)=tmp(i)+b(i)
  tmp(i+1)=c(i)
END DO
send A(2:101)=tmp(2:101)

```

【図 1 4】

## 本発明の最適化説明図

- (a)
- ```

DO i=1,100
  c(i)=A(i)+b(i)
  A(i+1)=c(i)
END DO

```
- (b)
- ```

c(1:100)=A(1:100)+b(1:100)
A(0:99)=c(1:100)

```
- (c)
- ```

receive tmp1(1:100)=A(1:100)
c(1:100)=tmp1(1:100)+b(1:100)
tmp2(0:99)=c(1:100)
send A(0:99)=tmp2(0:99)

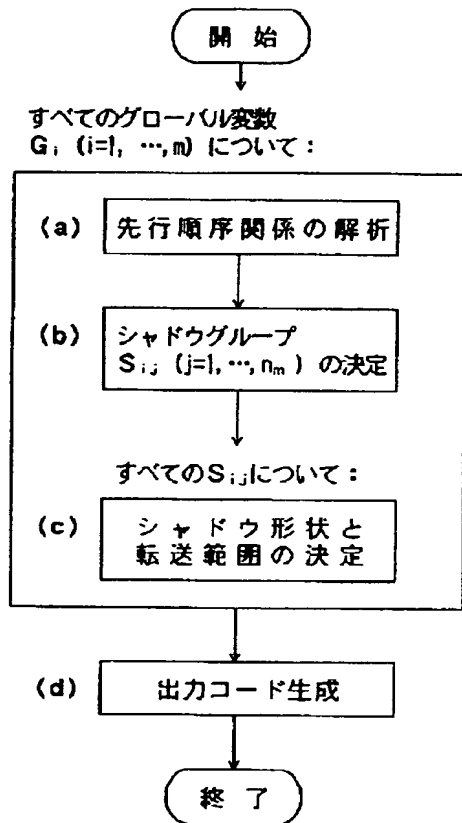
```
- (d)
- ```

DO i=1,100
  c(i)=A(i)+b(i)
  A(i+1)=c(i)
END DO

```

【図3】

## 実施例処理フローチャート



【図6】

## 出力コードの例

```

alloc a1(100,100)
alloc a2(100,170)

receive a1(1:100,1:100)=A(1:100,1:100)
receive a2(1:100,1:100)=A(1:100,31:130)

DO J=1,100
  DO I=1,100
    a1(I,J)=...
    -- = a1(J,I)
    -- = a2(I,J)
    a2(I,J+70)= --
    -- = a1(I,1)
  END DO
END DO

send A(1:100,1:100)=a1(1:100,1:100)
send A(1:100,101:200)=a2(1:100,71:170)

free a1
free a2
  
```

A (S1)  
A (S2)  
  
u1  
u2  
u3  
u4  
u5  
  
I (S1)  
I (S2)

【図10】

## 出力コードの例

```

alloc a(100,200)

receive a(1:100,1:100)=A(1:100,1:100)

DO J=1,100
  DO I=1,100
    a(I,J)= --
    -- = a(J,I)
    -- = a(I,J+30)
    a(I,J+100)= --
    -- = a(I,1)
  END DO
END DO

send A(1:100,1:200)=a(1:100,1:200)

free a
  
```

A (S1) U A (S2)  
  
u1  
u2  
u3  
u4  
u5  
  
I (S1) U I (S2)

【図9】

## 出力コードの例

```

alloc a1(100,100)
alloc a2(100,170)

receive a1(1:100,1:100)=A(1:100,1:100)
receive a2(1:100,1:70)=A(1:100,31:100)

DO J=1,100
  DO I=1,100
    a1(I,J)=...
    -- = a1(J,I)
    -- = a2(I,J)
    a2(I,J+70)= --
    -- = a1(I,1)
  END DO
END DO

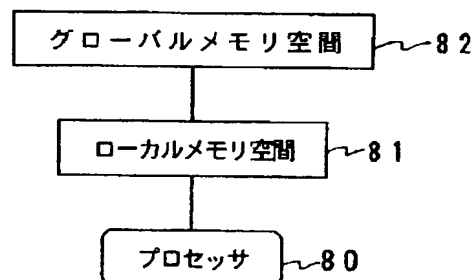
send A(1:100,1:100)=a1(1:100,1:100)
send A(1:100,101:200)=a2(1:100,71:170)

free a1
free a2
  
```

A (S1)  
A (S2)  
  
u1  
u2  
u3  
u4  
u5  
  
I (S1)  
I (S2)

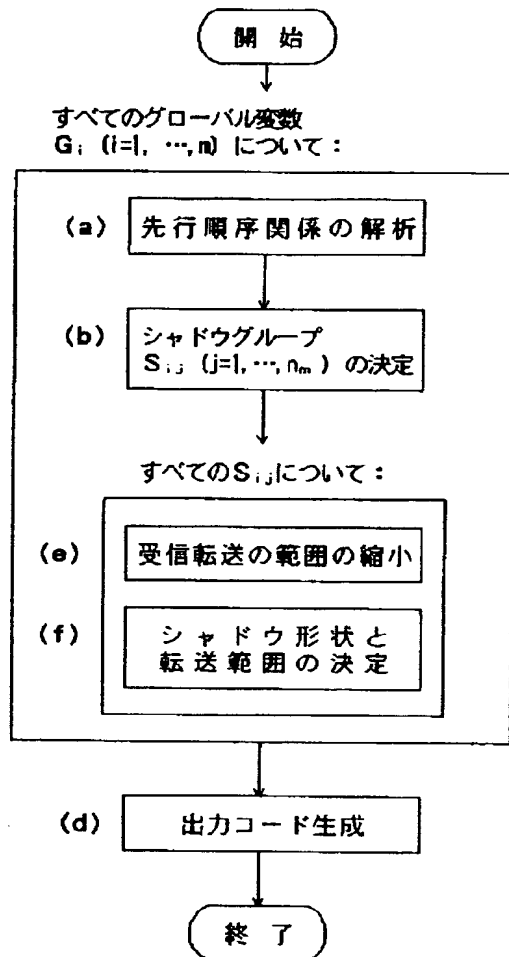
【図13】

## 2階層メモリのモデル説明図



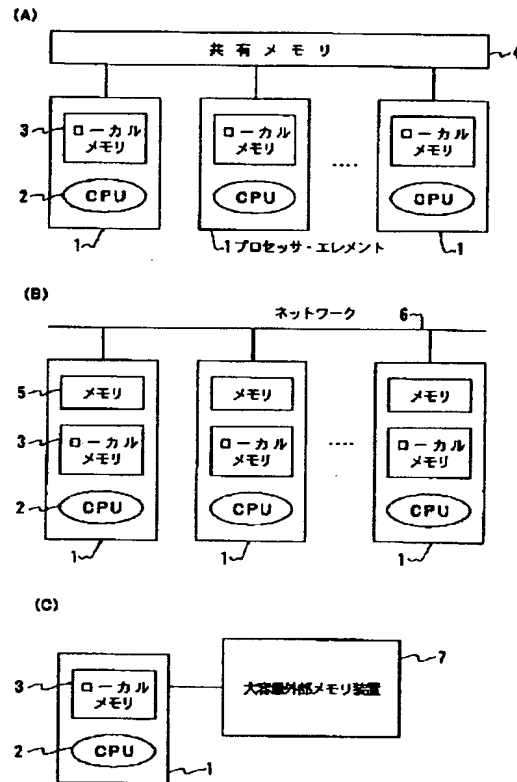
【図7】

## 実施例処理フローチャート



【図12】

## 本発明が適用可能なシステム構成の例



【図8】

受信転送範囲縮小のフローチャート

